A 2.5D APPROACH TO 360 PANORAMA VIDEO STABILIZATION

Lin-Chen Shen[⋆] *Tzu-Kuei Huang*[⋆] *Chu-Song Chen*[†] *Yung-Yu Chuang*[⋆]

[⋆]National Taiwan University [†]Academia Sinica

ABSTRACT

This paper presents a method for stabilizing both cylindrical and spherical panorama videos with a 360-degree field of view. We observe that rotation needs to be extremely smooth for 360 videos to maintain global motion coherency and avoid wobbling. Our method decouples the rotation from other motions and applies different strategies for smoothing them. The proposed approach is 2.5D as it estimates 3D rotations without involving 3D structure-from-motion methods. Therefore, it is more robust and can be performed in an incremental way. Experiments show that our method is effective in making steady 360 cylindrical/spherical videos.

Index Terms— 360 videos, video stabilization.

1. INTRODUCTION

Conventional cameras have very narrow fields of view. In many occasions, the user would like to record the scene with a much wider field of view, even with full 360°. In early days, one has to take multiple shots and assemble them together as a 360° image by stitching. Recently, many cameras with 360° field of view have emerged into the consumer market, such as Ricoh Theta and V.360. These cameras can record a full 360° spherical or cylindrical panorama image/video with a single shot. The recorded 360° imagery can serve as a more complete recording of important moments or the input to virtual reality systems. However, similar to conventional videos, casually captured 360° videos exhibit annoying jitter due to shaky motion of an unsteady hand-held camera or a head-mounted camera. This problem is even aggravated for a 360° video because of its extremely wide field of view. Thus, video stabilization, removing unwanted image perturbations due to unstable camera motions, is very important to 360° videos.

We have two main observations for 360° video stabilization. First, for stabilizing 360° videos, camera rotation needs to be very smooth to maintain global motion coherency and avoid wobbling. Second, although rotation needs to be heavily smooth, translation can be more tolerated. If they are coupled together, it is impossible to filter them differently. Thus, it is better to decouple them and apply different strategies for smoothing. From the above discussion, we know that rotation has significant influence on the smoothness of the video and needs to be handled carefully. Our method decouples the rotation from other motions, and handles both separately. Our method first estimates rotations and removes them completely. A mesh-based image warping approach is used to handle remaining jitters due to other motions, such as translations and parallax, after removing rotations. After compensating other motions, the rotation is restored by heavily smoothing the original rotations or set by the user interaction. Our method is 2.5D because it estimates 3D rotations without involving 3D structure-from-motion methods. It is more robust and can be performed in an online and streaming fashion.

This work was supported by MOST under grants 104-2628-E-002-003-MY3 and 107-2634-F-002-007.

2. RELATED WORK

Video stabilization. There are many video stabilization methods for conventional videos. These methods can be roughly categorized into 2D and 3D methods. 2D methods use a series of 2D transformations for smoothing camera motions [1, 2, 3, 4, 5]. 3D methods often rely on accurate 3D feature tracking and estimate the camera path in 3D [6, 7, 8, 9]. Although 3D approaches are capable of simulating better camera paths. they are generally less robust and often have to process the video as a whole. Thus, they have to be performed in an offline fashion and are not suitable for streaming applications. There are also several commercial tools for 360° video stabilization such as videostitch and autopano. They are however restricted to dedicated 360° cameras and are not generally applicable.

Mesh-based image warping. Our method uses mesh-based image warping for compensating jitters caused by non-rotation motions. Mesh-based image warping has been widely used to manipulate image structure for many applications such as as-rigid-as-possible image manipulation [10, 11], image retargeting [12, 13], video retargeting [14, 15, 16], image perspective manipulation [17], stereoscopic photo authoring [18], conventional video stabilization [6, 4, 8] and stereoscopic video stabilization [19].

3. CYLINDRICAL VIDEO STABILIZATION

There are two popular types of 360° videos: cylindrical and spherical. We will start with the cylindrical videos. Fig. 1 gives an overview to our approach for stabilizing 360 videos.

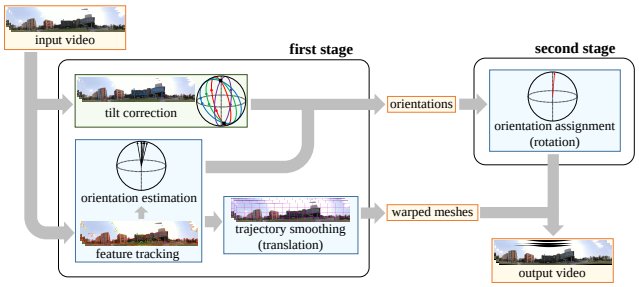

Fig. 1: The overview of our algorithm. Our method first estimates the orientation for each frame using the tracked features. After removing rotations, a mesh-based image warping method is used to handle remaining jitters. The new orientation for each frame can be determined by heavily smoothing or set by user interaction. The determined orientation is then restored to the frame to synthesize the output. Tilt correction is an optional step that can remove the unwanted camera tilt.

3.1. Camera projection model

We assume that the image plane is a cylinder whose radius is 1 as shown in the left side of Fig. 2. With this model, a cylindrical camera has two intrinsic parameters: the upward height h_u and the downward height h_d.With these parameters, given a pixel $p = (p_x, p_y)$ on the input image with the width W and the height H, the following mapping function finds its corresponding point on the projection cylinder:

$$\tilde{\phi}(p) = \begin{bmatrix} \sin\left(2\pi\frac{p_x}{W}\right) \\ \frac{p_y}{H}(h_u + h_d) - h_d \\ \cos\left(2\pi\frac{p_x}{W}\right) \end{bmatrix}. \quad (1)$$

By normalizing the point's coordinate, $\phi(p) = \frac{\tilde{\phi}(p)}{||\tilde{\phi}(p)||}$, we obtain the 3D direction that the pixel is meant to represent. The mapping function $\phi(p)$ takes a pixel p on the input image and finds the direction it represents. Its inverse function $\phi^{-1}(v)$ does the inverse by taking a 3D direction v and finding its corresponding pixel on the image plane.

3.2. Orientation estimation

Our method uses the KLT tracker [20] for tracking features along the input video. The results are a set of feature trajectories. For the t-th frame, let $\mathbf{P}^t$ be the set of features that we have their correspondences in the next frame. For a feature $p_k^t \in \mathbf{P}^t$, we denote its correspondence in the next frame as p_k^{t+1}. The motion of features between frames are caused by the camera motion including rotation and translation. As discussed above, we assume that the camera rotation has much more significant impact than translation. Therefore, we find the rotation between two neighboring frames by projecting features onto the unit sphere using ϕ and finding the rotation matrix which minimizes their differences, that is,

$$\mathbf{R}^{t,t+1} = \arg\min_{\mathbf{R}} \sum_{p_k^t \in \mathbf{P}^t} \left\| \mathbf{R}\phi\left(p_k^t\right) - \phi\left(p_k^{t+1}\right) \right\|^2. \quad (2)$$

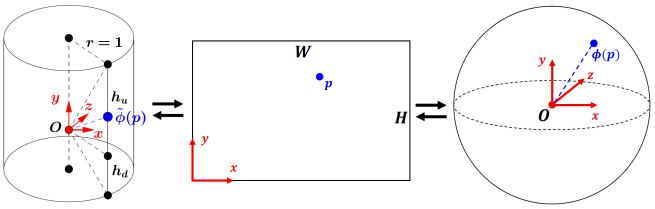

Fig. 2: Left: the cylindrical camera model with parameters h_u and h_d. Right: the mapping function from the input equirectangular coordinate to the unit sphere.

The 3D rotation matrix $\mathbf{R}^{t,t+1}$ can be solved by Kabsch algorithm [21]. This way, we obtain the rotations between adjacent frames. By matrix multiplication, we can obtain the rotation matrix $\mathbf{R}^{t,s}$ between any two frames t and s, which rotates the s-th frame to align with the t-th frame.

The orientation of a frame can be represented by the up direction $\mathbf{u}$ and the front direction $\mathbf{f}$. Let $\mathbf{u}^t$ and $\mathbf{f}^t$ be the up and front directions for the t-th frame. The rotation matrix $\mathbf{R}^{t,0}$ reveals the orientation of the frame t related to the reference frame with the following relationships: $\mathbf{u}^t = \mathbf{R}^{t,0}\mathbf{u}^0, \mathbf{f}^t = \mathbf{R}^{t,0}\mathbf{f}^0$. With these vectors, we have the orientation of each frame related to the reference frame.

3.3. Trajectory smoothing

After obtaining the orientation of each frame, the basic idea for stabilization is to first remove the rotation between two frames and employ mesh-based image warping [4] for compensating the remaining high-frequency jitters caused by camera translation and parallax. Our method uses a quad mesh to guide the deformation of each frame for compensating motion jitters. Assume that each mesh has R rows and C columns. Let $\mathbf{V}^t = \{v_{c,r} | c = 0..C, r = 0..R\}$ represent the vertices of the mesh for the t-th frame and $\mathbf{V} = \{\mathbf{V}^t | t = 1..T\}$. The goal of mesh-based video stabilization is to solve for the set of variables $\mathbf{V}$ so that the video looks stabilized and natural after image deformation guided by the deformed meshes. The objective function contains two terms: a data term and a shape-preserving term.

The data term $\mathbf{E}_d$ is designed to ensure that the video looks stabilized by smoothing feature trajectories. Let p_k^t be the pixel coordinate of the k-th feature in the t-th frame. Assume that it locates within the quad (c_k^t, r_k^t) where c_k^t and r_k^t are column and row indices respectively. The feature location is related to the vertex positions by $p_k^t = {\Omega_k^t}^T V_k^t$ where V_k^t is the matrix encoding the four corners of the quad and Ω_k^t is the vector with bilinear weights, that is,

$$V_k^t = \begin{bmatrix} v^t_{c_k^t, r_k^t} \\ v^t_{c_k^t+1, r_k^t} \\ v^t_{c_k^t+1, r_k^t+1} \\ v^t_{c_k^t, r_k^t+1} \end{bmatrix}, \Omega_k^t = \begin{bmatrix} \omega_k^t(0,0) \\ \omega_k^t(1,0) \\ \omega_k^t(1,1) \\ \omega_k^t(0,1) \end{bmatrix}. \quad (3)$$

The data term $\mathbf{E}_d(\mathbf{V})$ is defined as:

$$\sum_k \sum_{t,t'} G_1(t-t') \|\Omega_k^{t\,T} V_k^t - \phi^{-1}(\mathbf{R}^{t',t}\phi(\Omega_k^{t'\,T} V_k^{t'}))\|^2, \quad (4)$$

where t and t' are two frames where the k-th feature appears and G_1 is a Gaussian function which assigns more weights to closer frames. Note that $\phi^{-1}(\mathbf{R}^{t',t}\phi(q))$ transforms a pixel q to a 3D point on the unit sphere, rotates it and maps it back to the 2D image plane. Its effect is to remove the rotation between the two frames. The term is used to remove remaining jitter after removing rotations.

A shape-preserving term $\mathbf{E}_s$ is used for regularization so that regions with less or no feature can be better constrained. Here, we use a design similar to Carroll et al.'s method for perspective manipulation [17] and define the term $\mathbf{E}_s(\mathbf{V})$ as

$$\sum_t \sum_{c,r} (\left\|2v_{c,r}^t - v_{c+1,r}^t - v_{c-1,r}^t\right\|^2 + \left\|2v_{c,r}^t - v_{c,r+1}^t - v_{c,r-1}^t\right\|^2). \quad (5)$$

The objective function $\mathbf{E}(\mathbf{V})$ combines the data term and the shape-preserving term as $\mathbf{E}(\mathbf{V}) = \mathbf{E}_d(\mathbf{V}) + \lambda \mathbf{E}_s(\mathbf{V})$, where λ balances these two terms and $\lambda = 0.2$ in all experiments. $\mathbf{E}(\mathbf{V})$ is linear and solved effectively with Jacobi solver [22].

3.4. Orientation assignment

This stage attempts to restore proper camera rotations which can be set either according to users' interaction (for interactive players or VR applications) or by heavily smoothing the camera rotations in the input video. This section describes a method for the latter. The orientation can be uniquely specified by its up direction and front direction. We first obtain the smooth up direction $\tilde{\mathbf{u}}^t$ for each frame by averaging with neighboring frames with a Gaussian filter G_2:

$$\tilde{\mathbf{u}}^t = \sum_s G_2(s-t)\mathbf{u}^s. \quad (6)$$

After finding the smooth up direction $\tilde{\mathbf{u}}^t$, the next step is to find the smooth front direction $\tilde{\mathbf{f}}^t$ for each frame. We determine a rotation matrix by

$$\bar{\mathbf{R}}^t = \arg\min_{\mathbf{R}} \left(\|\mathbf{R}\tilde{\mathbf{u}}^{t-1} - \tilde{\mathbf{u}}^t\|^2 + \|\mathbf{R}(\tilde{\mathbf{u}}^{t-1} \times \tilde{\mathbf{u}}^t) - (\tilde{\mathbf{u}}^{t-1} \times \tilde{\mathbf{u}}^t)\|^2\right). \quad (7)$$

and use it to rotate $\tilde{\mathbf{f}}^{t-1}$ for obtaining $\tilde{\mathbf{f}}^t$, i.e., $\tilde{\mathbf{f}}^t = \bar{\mathbf{R}}^t \tilde{\mathbf{f}}^{t-1}$.

Once we have determined the target up and front directions for each frame, $\tilde{\mathbf{u}}^t$ and $\tilde{\mathbf{f}}^t$, the next step is to find a proper rotation matrix to bring the original up vector $\mathbf{u}^t$ to the target up vector $\tilde{\mathbf{u}}^t$ and $\mathbf{f}^t$ to $\tilde{\mathbf{f}}^t$ for the front vector. The rotation matrix $\tilde{\mathbf{R}}^t$ can be found by

$$\tilde{\mathbf{R}}^t = \arg\min_{\mathbf{R}} \left(\|\mathbf{R}\tilde{\mathbf{u}}^t - \mathbf{u}^t\|^2 + \|\mathbf{R}\tilde{\mathbf{f}}^t - \mathbf{f}^t\|^2\right). \quad (8)$$

Next, we apply the rotation on the warped mesh specified by $\mathbf{V}^t$. It is carried out by using the following equation.

$$\tilde{\mathbf{V}}^t = \phi^{-1}(\tilde{\mathbf{R}}^t \phi(\mathbf{V}^t)). \quad (9)$$

The result $\tilde{\mathbf{V}}^t$ specifies the warped mesh that rotates the previous warped mesh according to $\tilde{\mathbf{R}}^t$.

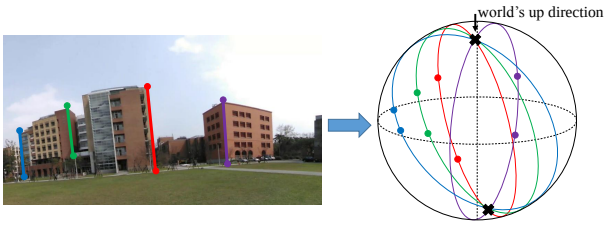

Fig. 3: Estimation of the world's up direction. Each detected line segment in the image plane corresponds to a great circle in the unit sphere. The great circles of vertical scene lines should intersect on the north pole, the world's up direction.

3.5. Tilt correction

The method in the previous section finds the orientations related to the reference frame. It however cannot find the orientation with respect to the world. Thus, if the camera has a tilt angle, the tilt would remain in the stabilized video. We describe a method for finding the camera tilt and its correction. We first use the LSD line detector [23] for finding line segments in frames. In most scenes, vertical and horizontal lines dominate. We care more about the vertical lines and remove lines whose angles to the horizon are less than $45°$. We then detect vertical scene lines by finding the dominant direction of the remaining lines.

Tilt detection. Assume that e_i^1 and e_i^2 are the two end pixels of the i-th line segment. As shown in Fig. 3, by mapping the end pixels to the unit sphere, each line segment corresponds to a great circle on the unit sphere. For vertical scene lines, their great circles should intersect on the north pole (the world's up direction). Thus, we can find the world's up direction by finding the intersection that most remaining lines agree.

For each line segment (e_i^1, e_i^2), we find its left direction $\mathbf{l}_i$ perpendicular to its great circle by $\mathbf{l}_i = \phi(e_i^1) \times \phi(e_i^2)$. The intersection of the great circles for a set of lines should be perpendicular to their left directions. Thus, the intersection $\bar{\mathbf{w}}$ for a set of lines L can be found by

$$\bar{\mathbf{w}} = \arg\min_{\mathbf{w}} \sum_{l_i \in L} \|\mathbf{l}_i{}^T \mathbf{w}\|^2 \text{ subject to } \|\mathbf{w}\| = 1. \quad (10)$$

It can be solved by SVD. Note that there could be non-vertical lines in the remaining lines and they need to be detected and excluded from the calculation of the intersection. Since vertical scene lines should dominate, RANSAC can be used for detecting non-vertical lines and finding the world's up direction $\bar{\mathbf{w}}^t$ for the t-th frame. Finally we convert the coordinate system from the current frame to the reference frame: $\mathbf{w}^t = \mathbf{R}^{t,0}\bar{\mathbf{w}}^t$.

Tilt correction. Ideally, the world's up directions estimated from frames should be identical. However, in practice, they could be different due to noise, numerical inaccuracy and errors in RANSAC. Thus, to be more robust and stable, the Gaussian filter G_2 is applied to smooth estimations of neighboring frames and the smooth version is taken as the world's

up direction for the current frame:

$$\tilde{\mathbf{u}}^t = \sum_s G_2(s-t)\mathbf{w}^s. \quad (11)$$

Once the up direction is determined for the current frame, the calculation of the front direction and the mesh warping is the same as described in Section 3.4.

4. SPHERICAL VIDEO STABILIZATION

The pipeline described in Section 3 can also be used for stabilizing spherical 360° videos. The components for orientation estimation and orientation assignment are identical. The main differences lie on the camera projection model and the trajectory smoothing method.

Camera projection model. The input frames use the equirectangular projection. The image coordinate (x, y) of a pixel p corresponds to its longitude and latitude. We can easily find its corresponding position on the 3D unit sphere (the right of Fig. 2). Given a pixel $p = (p_x, p_y)$, the mapping function from p to its corresponding point on the 3D unit sphere is:

$$\phi(p) = \begin{bmatrix} \sin(\pi \cdot \frac{H-p_y}{H}) \cos(2\pi \cdot \frac{p_x}{W}) \\ \cos(\pi \cdot \frac{H-p_y}{H}) \\ \sin(\pi \cdot \frac{H-p_y}{H}) \sin(2\pi \cdot \frac{p_x}{W}) \end{bmatrix}. \quad (12)$$

With the projection model, the orientation estimation method described in Section 3.2 can be used for estimating rotations.

Trajectory smoothing. Different from the cylindrical case, a sphere cannot be easily unfolded onto a plane like a cylinder. Thus, instead of a quad mesh on a plane, we use an icosphere mesh on a sphere to guide the warping. Our trajectory smoothing method directly works on the isosphere in 3D. Let p_k^t be the pixel coordinate of the k-th feature in the t-th frame. Assume that it locates within the triangular face with three vertices $v_{k,1}^t$, $v_{k,2}^t$ and $v_{k,3}^t$. We can relate the location of p_k^t with the three vertices with barycentric weights $\omega_k^t(1)$, $\omega_k^t(2)$, $\omega_k^t(3)$ as $p_k^t = {\Omega_k^t}^T V_k^t$ where

$$V_k^t = \begin{bmatrix} v_{k,1}^t \\ v_{k,2}^t \\ v_{k,3}^t \end{bmatrix}, \Omega_k^t = \begin{bmatrix} \omega_k^t(1) \\ \omega_k^t(2) \\ \omega_k^t(3) \end{bmatrix}. \quad (13)$$

The data term is defined as:

$$\mathbf{E}_d(\mathbf{V}) = \sum_k \sum_{t,t'} G_1(t-t') \|{\Omega_k^t}^T V_k^t - \mathbf{R}^{t',t} {\Omega_k^{t'}}^T V_k^{t'}\|^2. \quad (14)$$

The shape-preserving term is designed for maintaining the straightness of cell edges and assigning constraints for featureless regions. There are two types of vertices in the icosphere. The first type is the base vertex $\mathbf{V}_b$, the vertices of the 20-face basic isosphere. Each base vertex has five neighbors. The second type is the subdivision vertex $\mathbf{V}_s$, the vertices generated by subdivision and each of them has six neighbors. Thus, the shape preserving term for base vertices is:

$$\mathbf{E}_s^b(\mathbf{V}) = \sum_{v_k^t \in \mathbf{V}_b} \|v_k^t - \frac{\sum_{v' \in N_b(v_k^t)} v'}{5}\|^2, \quad (15)$$

where $N_b(v_k^t)$ denotes the five neighbors of a base vertex v_k^t. The shape-preserving term for subdivision vertices is:

$$\mathbf{E}_s^s(\mathbf{V}) = \sum_{v_k^t \in \mathbf{V}_s} \|v_k^t - \frac{\sum_{v' \in N_s(v_k^t)} v'}{2}\|^2, \quad (16)$$

where $N_s(v_k^t)$ denotes the two neighbors along the edge where v_k^t was generated at. Note that the vertices should remain on the sphere after optimization. Thus, we need to add the constraint $\|v\| = 1,\ \forall\ v \in \mathbf{V}$ into the objective function with Lagrange multipliers:

$$\mathbf{E}_l(\mathbf{V}, \{\gamma\}) = \sum_{v_k^t \in \mathbf{V}} \gamma_k^t \cdot (\|v_k^t\| - 1), \quad (17)$$

where γ_k^t is the Lagrange multiplier corresponding to v_k^t. The objective function $\mathbf{E}(\mathbf{V}, \{\gamma\})$ combines the above terms:

$$\mathbf{E}(\mathbf{V}, \{\gamma\}) = \mathbf{E}_d(\mathbf{V}) + \lambda(\mathbf{E}_s^b(\mathbf{V}) + \mathbf{E}_s^s(\mathbf{V})) + \mathbf{E}_l(\mathbf{V}, \{\gamma\}). \quad (18)$$

5. EXPERIMENTS AND RESULTS

All experiments were executed on a machine with an Intel i7 3.4GHz processor and 8GB memory. The σ values of G_1 and G_2 are respectively set to 3.16 and 83.33 for all experiments. The computation was carried out in an online fashion with a sliding window. That is, after reading initially a few frames, say 60 frames, our method produces a stabilized frame for each frame. A 3D approach would require reading the whole video and is not suitable for online processing.

The input cylindrical videos were taken by a V.360 camera. The resolution is 3240×540. The grid size of the mesh is 50×15. On average, it took around 300ms for processing a frame including tilt correction. Feature detection, trajectory smoothing and tilt correction took 110ms, 160ms and 30ms respectively. The input spherical videos were taken by a Ricoh Theta S camera. The resolution is 1920×960. We subdivided the basic icosphere for 3 times, leading to a 1280-face isosphere. The other settings are the same as the cylindrical case. Since the results are best viewed in the form of videos, please refer to the accompanying video for results.

6. CONCLUSION

The paper presents a 2.5D approach for 360° panorama video stabilization. The method deals with both cylindrical and spherical videos. The main advantage of the proposed method is that it estimates the 3D rotation without resorting to structure from motion. This way, the proposed method can be executed in an online fashion and can be more robust.

7. REFERENCES

[1] Ken-Yi Lee, Yung-Yu Chuang, Bing-Yu Chen, and Ming Ouhyoung, "Video stabilization using robust feature trajectories," in *Proceedings of IEEE Conference on Computer Vision (ICCV 2009)*, 2009, pp. 1397–1404.

[2] S. Battiato, A. R. Bruna, and G. Puglisi, "A robust block-based image/video registration approach for mobile imaging devices," *IEEE Transactions on Multimedia*, vol. 12, no. 7, pp. 622–635, 2010.

[3] Matthias Grundmann, Vivek Kwatra, and Irfan Essa, "Auto-directed video stabilization with robust L1 optimal camera paths," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2011, pp. 225–232.

[4] Shuaicheng Liu, Lu Yuan, Ping Tan, and Jian Sun, "Bundled camera paths for video stabilization," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, pp. 78, 2013.

[5] Shuaicheng Liu, Lu Yuan, Ping Tan, and Jian Sun, "Steadyflow: Spatially smooth optical flow for video stabilization," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2014, pp. 4209–4216.

[6] Feng Liu, Michael Gleicher, Hailin Jin, and Aseem Agarwala, "Content-preserving warps for 3d video stabilization," in *ACM Transactions on Graphics (TOG)*, 2009, vol. 28, p. 44.

[7] Amit Goldstein and Raanan Fattal, "Video stabilization using epipolar geometry," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 5, pp. 126, 2012.

[8] Yu-Shuen Wang, Feng Liu, Pu-Sheng Hsu, and Tong-Yee Lee, "Spatially and temporally optimized video stabilization," *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 19, no. 8, pp. 1354–1361, 2013.

[9] Zihan Zhou, Hailin Jin, and Yi Ma, "Plane-based content preserving warps for video stabilization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2299–2306.

[10] Takeo Igarashi, Tomer Moscovich, and John F Hughes, "As-rigid-as-possible shape manipulation," *ACM transactions on Graphics (TOG)*, vol. 24, no. 3, pp. 1134–1141, 2005.

[11] Scott Schaefer, Travis McPhail, and Joe Warren, "Image deformation using moving least squares," in *ACM transactions on graphics (TOG)*, 2006, vol. 25, pp. 533–540.

[12] Yu-Shuen Wang, Chiew-Lan Tai, Olga Sorkine, and Tong-Yee Lee, "Optimized scale-and-stretch for image resizing," *ACM Transactions on Graphics (TOG)*, vol. 27, no. 5, pp. 118, 2008.

[13] S. S. Lin, I. C. Yeh, C. H. Lin, and T. Y. Lee, "Patch-based image warping for content-aware retargeting," *IEEE Transactions on Multimedia*, vol. 15, no. 2, pp. 359–368, 2013.

[14] Yu-Shuen Wang, Hongbo Fu, Olga Sorkine, Tong-Yee Lee, and Hans-Peter Seidel, "Motion-aware temporal coherence for video resizing," *ACM Transactions on Graphics (TOG)*, vol. 28, no. 5, pp. 127, 2009.

[15] Z. Qu, J. Wang, M. Xu, and H. Lu, "Context-aware video retargeting via graph model," *IEEE Transactions on Multimedia*, vol. 15, no. 7, pp. 1677–1687, 2013.

[16] D. Y. Chen and Y. S. Luo, "Preserving motion-tolerant contextual visual saliency for video resizing," *IEEE Transactions on Multimedia*, vol. 15, no. 7, pp. 1616–1627, 2013.

[17] Robert Carroll, Aseem Agarwala, and Maneesh Agrawala, "Image warps for artistic perspective manipulation," in *ACM Transactions on Graphics (TOG)*, 2010, vol. 29, p. 127.

[18] F. Liu, Y. Niu, and H. Jin, "Casual stereoscopic photo authoring," *IEEE Transactions on Multimedia*, vol. 15, no. 1, pp. 129–140, 2013.

[19] Chun-Wei Liu, Tz-Huan Huang, Ming-Hsu Chang, Ken-Yi Lee, Chia-Kai Liang, and Yung-Yu Chuang, "3D cinematography principles and their applications to stereoscopic media processing," in *Proceedings of ACM International Conference on Multimedia*, November 2011, pp. 253–262.

[20] Bruce D Lucas, Takeo Kanade, et al., "An iterative image registration technique with an application to stereo vision.," in *IJCAI*, 1981, vol. 81, pp. 674–679.

[21] Wolfgang Kabsch, "A solution for the best rotation to relate two sets of vectors," *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, vol. 32, no. 5, pp. 922–923, 1976.

[22] IN Bronshtein, KA Semendyayev, and KA Kirsch, "Handbook of mathematics," 1997.

[23] Rafael Grompone von Gioi, Jérémie Jakubowicz, Jean-Michel Morel, and Gregory Randall, "LSD: a line segment detector," *Image Processing On Line*, vol. 2, pp. 35–55, 2012.